# Dual Motor Controller CAN ICD

Rev 1.0

# 1. Introduction

This document is an interface control document defining the CAN messages supported by the CAN Controlled Dual Closed-Loop Motor Controller. This system uses standard 11-bit CAN identifiers, with a bus speed of 1Mbps. All multi-byte values are sent as little endian.

# 2. CAN Frame Identifier Breakdown

The most significant bit indicates a message type: either a command or info message. The next 3 most significant bits indicate a message class. The next 3 most significant bits indicate a message index. Finally, the 4 least significant bits indicate what device should be commanded (in case of a command message), or is sending the informational message (in case of an info message.)

| **Bits** | x | xxx | xxx | xxxx |
|---|---|---|---|---|
| **Use** | Message type:<br>1 - command<br>0 - info | Message class | Message index | Message device index |

# 3. CAN Messages List

## Command Messages

### Message Class 0: Command control

#### Class 0, Index 0: Command position

| Length | 5 bytes |
|---|---|
| Value 0: Motor index | 1 byte: uint8_t, the motor to be commanded |

| | |
|---|---|
| Value 1: Position command | 4 bytes: float32, the position to be commanded in degrees of the output shaft (gear ratio accounted for) |

### Class 0, Index 1: Command speed

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index | 1 byte: uint8_t, the motor to be commanded |
| Value 1: Speed command | 4 bytes: int32_t, the position to be commanded in degrees/s of the output shaft (gear ratio accounted for) |

### Class 0, Index 2: Command current

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index | 1 byte: uint8_t, the motor to be commanded |
| Value 1: Current command | 4 bytes: int32_t, the current in mA to be commanded |

### Class 0, Index 3: Command motion primitive

| | |
|---|---|
| Length | 7 bytes |
| Value 0: Primitive index | 1 byte: uint8_t, the motion primitive to be used |
| Value 1: Primitive period | 2 bytes: int16_t, the motion primitive period in ms |
| Value 2: Primitive time offset | 2 bytes: int16_t, the time offset of the motion primitive in ms |
| Value 3: Invert | 1 byte: uint8_t, inverts motion in joint space if value > 0 |
| Value 4: Time reversal | 1 byte: uint8_t, reverses motion in time if value > 0 |

### Class 0, Index 4: Command duty

| | |
|---|---|
| Length | 5 bytes |

| | |
|---|---|
| Value 0: Motor index | 1 byte: uint8_t, the motor to be commanded |
| Value 1: Duty command | 4 bytes: float32, the open-loop duty cycle command to be set in [-1,+1] |

## Message Class 1: Command time

### Class 1, Index 0: Command sync time

| | |
|---|---|
| Length | 4 bytes |
| Value 0: External time | 4 bytes: uint32_t, the time of main controller in ms |

## Message Class 2: Command set parameter

### Class 2, Index 0: Command motion primitive scaling

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Primitive index | 1 byte: uint8_t, the motion primitive to be used |
| Value 1: Primitive x offset | 1 byte: int8_t, x-axis offset of primitive in mm |
| Value 2: Primitive y offset | 1 byte: int8_t, y-axis offset of primitive in mm |
| Value 3: Primitive x scale | 1 byte: uint8_t, scale of primitive in x, percentage (e.g. value of 100 does no scaling) |
| Value 4: Primitive y scale | 1 byte: uint8_t, scale of primitive in y, percentage (e.g. value of 100 does no scaling) |

### Class 2, Index 1: Command pd and min/max parameters

| | |
|---|---|
| Length | 8 bytes |
| Value 0: Motor index, control type | 1 byte: uint8_t, lsb is motor index, next 2 lsbs are 0-position, 1-speed, 2-current |
| Value 1: kp | 2 bytes: int16_t, kp proportional gain parameter * 100 (eg. for kp=75.0, send 7500) |

| | |
|---|---|
| Value 2: kd | 2 byte: int16_t, kd proportional gain parameter * 100 |
| Value 3: speed filter | 1 byte: uint8_t, Speed is filtered as:<br>speed = alpha * last_speed + (1.0-alpha) * new_speed. This value is alpha * 100. (e.g. for alpha = 0.95, send 95). Must be <100 |
| Value 4: Command max | 1 byte: uint8_t, command max in percentage of max. Must be <= 100 |
| Value 5: Command min | 1 byte: uint8_t, command min in percentage of min. Must be <= 100 |

### Class 2, Index 2: Command integral gain param

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index, control type | 1 byte: uint8_t, lsb is motor index, next 2 lsbs are 0-position, 1-speed, 2-current |
| Value 1: ki | 4 byte: float32, ki integral gain parameter |

### Class 2, Index 3: Command max integral windup

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index, control type | 1 byte: uint8_t, lsb is motor index, next 2 lsbs are 0-position, 1-speed, 2-current |
| Value 1: max windup | 4 byte: float32, maximum integral windup |

### Class 2, Index 4: Command min integral windup

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index, control type | 1 byte: uint8_t, lsb is motor index, next 2 lsbs are 0-position, 1-speed, 2-current |
| Value 1: min windup | 4 byte: float32, minimum integral windup |

#### Class 2, Index 5: Command ticks per output shaft revolution

| | |
|---|---|
| Length | 5 bytes |
| Value 0: Motor index | 1 byte: uint8_t, motor index |
| Value 1: ticks per rev | 4 byte: int32_t, Number of ticks per revolution of output shaft. Must incorporate both CPR and gearbox ratio. |

#### Class 2, Index 6: Command motion primitive keyframe

| | |
|---|---|
| Length | 8 bytes |
| Value 0: Primitive  index | 1 byte: uint8_t, primitive index |
| Value 1: Keyframe index | 1 byte: uint8_t, keyframe index |
| Value 2: Keyframe X | 2 bytes: int16_t, position in mm |
| Value 3:: Keyframe Y | 2 bytes: int16_t, position in mm |
| Value 4: Keyframe time part | 2 bytes: uint16_t, part of time period, multiplied by 2^16-1. (e.g. for t_part = 0.5, send 32768) |

### Message Class 3: Command zero position

#### Class 3, Index ANY: Command zero position

| | |
|---|---|
| Length | 1 byte |
| Value 0: Motor index | 1 byte: uint8_t, the motor to have its position zeroed |

## Info Messages

### Message Class 0: Info motor telemetry

#### Class 0, Index 0: Info position

| | |
|---|---|
| Length | 8 bytes |

| | |
|---|---|
| Value 0: Motor 0 position | 4 bytes: int32_t, the ticks count of motor |
| Value 1: Motor 1 position | 4 bytes: int32_t, the ticks count of motor |

### Class 0, Index 1: Info current

| | |
|---|---|
| Length | 8 bytes |
| Value 0: Motor 0 current | 4 bytes: int32_t, the current of motor in mA |
| Value 1: Motor 1 current | 4 bytes: int32_t, the current of motor in mA |

### Class 0, Index 2: Info speed

| | |
|---|---|
| Length | 8 bytes |
| Value 0: Motor 0 speed | 4 bytes: int32_t, the speed of motor in output shaft deg/s |
| Value 1: Motor 1 speed | 4 bytes: int32_t, the speed of motor in output shaft deg/s |